

**CURSO DE JAVA CON JDBC**

# **EJERCICIO**

## **EJERCICIO CAPA DE DATOS CON JDBC**



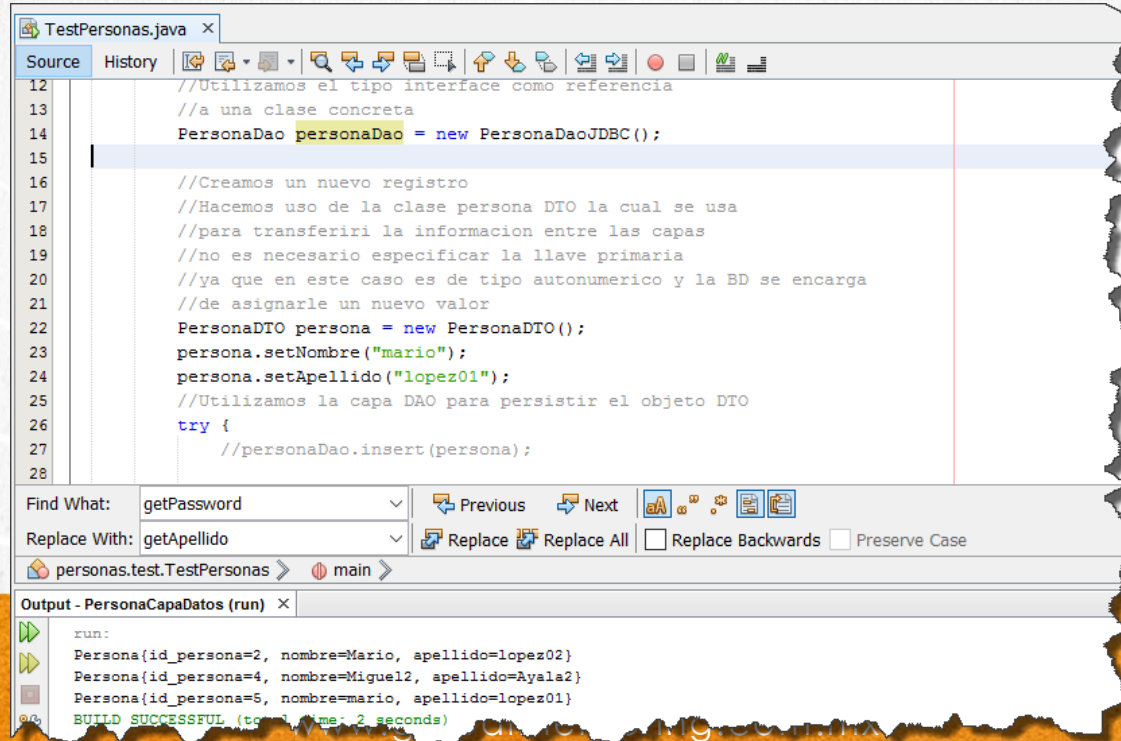
Experiencia y Conocimiento para tu vida

**CURSO DE JAVA CON JDBC**

[www.globalmentoring.com.mx](http://www.globalmentoring.com.mx)

# OBJETIVO DEL EJERCICIO

Crear un programa para crear una capa de datos lógica utilizando JDBC. Al finalizar deberemos observar lo siguiente:



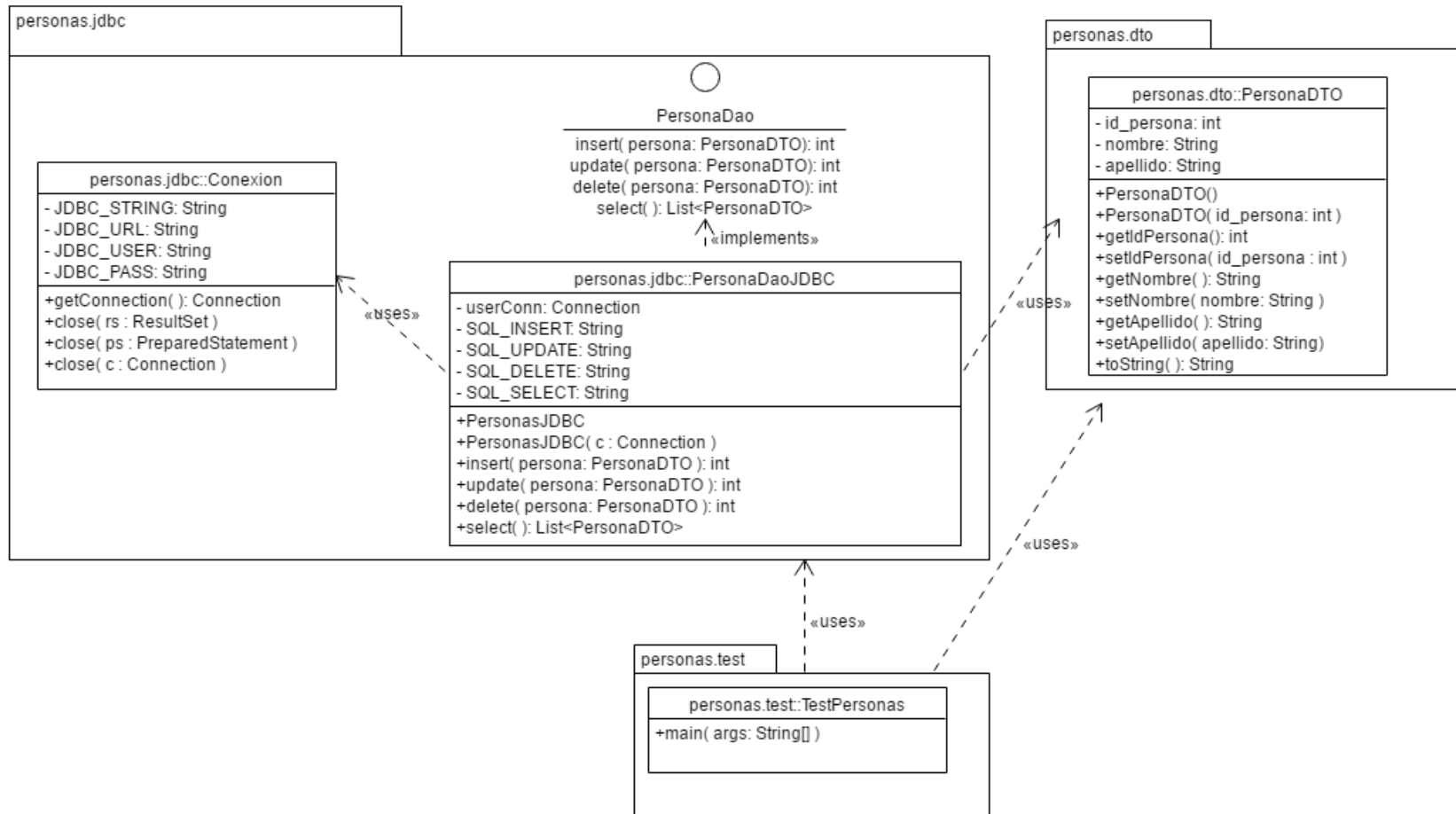
The screenshot shows an IDE window titled 'TestPersonas.java'. The code is as follows:

```
12 //Utilizamos el tipo interface como referencia
13 //a una clase concreta
14 PersonaDao personaDao = new PersonaDaoJDBC();
15
16 //Creamos un nuevo registro
17 //Hacemos uso de la clase persona DTO la cual se usa
18 //para transferir la informacion entre las capas
19 //no es necesario especificar la llave primaria
20 //ya que en este caso es de tipo autonumerico y la BD se encarga
21 //de asignarle un nuevo valor
22 PersonaDTO persona = new PersonaDTO();
23 persona.setNombre("mario");
24 persona.setApellido("lopez01");
25 //Utilizamos la capa DAO para persistir el objeto DTO
26 try {
27     //personaDao.insert(persona);
28 }
```

The IDE interface includes a toolbar, a 'Find What' field with 'getPassword', a 'Replace With' field with 'getApellido', and buttons for 'Previous', 'Next', 'Replace', 'Replace All', 'Replace Backwards', and 'Preserve Case'. The 'Output - PersonaCapaDatos (run)' window at the bottom shows the following output:

```
run:
Persona{id_persona=2, nombre=Mario, apellido=lopez02}
Persona{id_persona=4, nombre=Miguel2, apellido=Ayala2}
Persona{id_persona=5, nombre=mario, apellido=lopez01}
BUILD SUCCESSFUL (total time: 2 seconds)
```

# DIAGRAMA DE CLASES



# PASO 1. CREACIÓN DEL PROYECTO

Vamos a crear el proyecto:

New Java Application

**Steps**

1. Choose Project
2. **Name and Location**

**Name and Location**

Project Name: PersonaCapaDatos

Project Location: C:\Cursos\JDBC\Leccion06 Browse...

Project Folder: C:\Cursos\JDBC\Leccion06\PersonaCapaDatos

☐ Use Dedicated Folder for Storing Libraries

Libraries Folder: Browse...

Different users and projects can share the same compilation libraries (see Help for details).

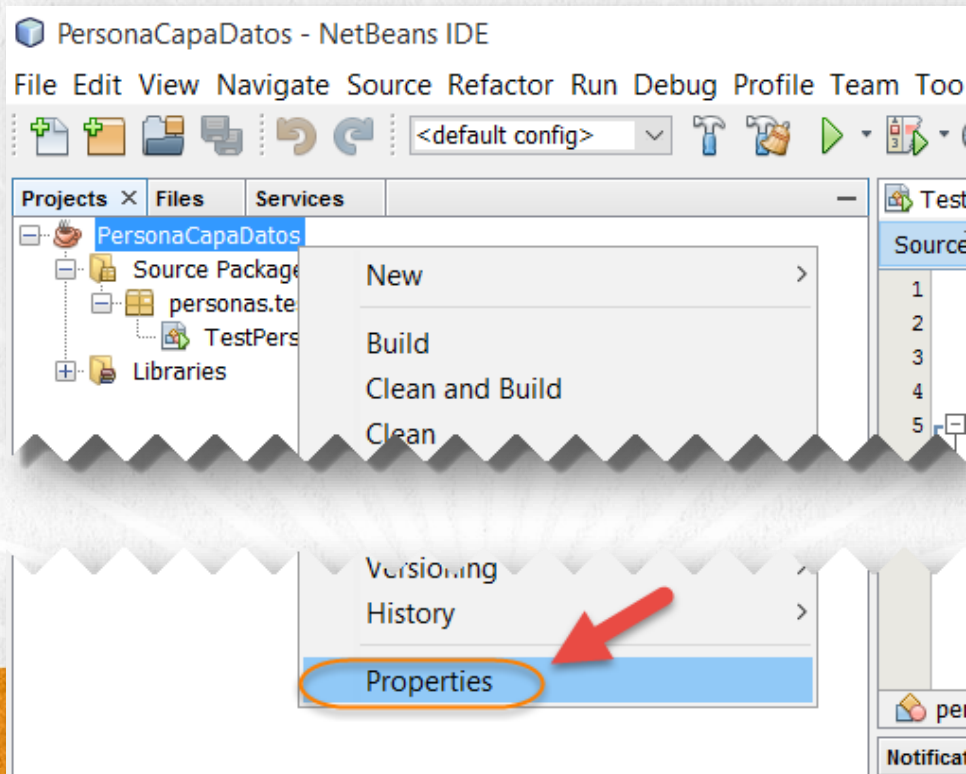
☒ Create Main Class personas.test.TestPersonas

< Back Next > Finish Cancel Help



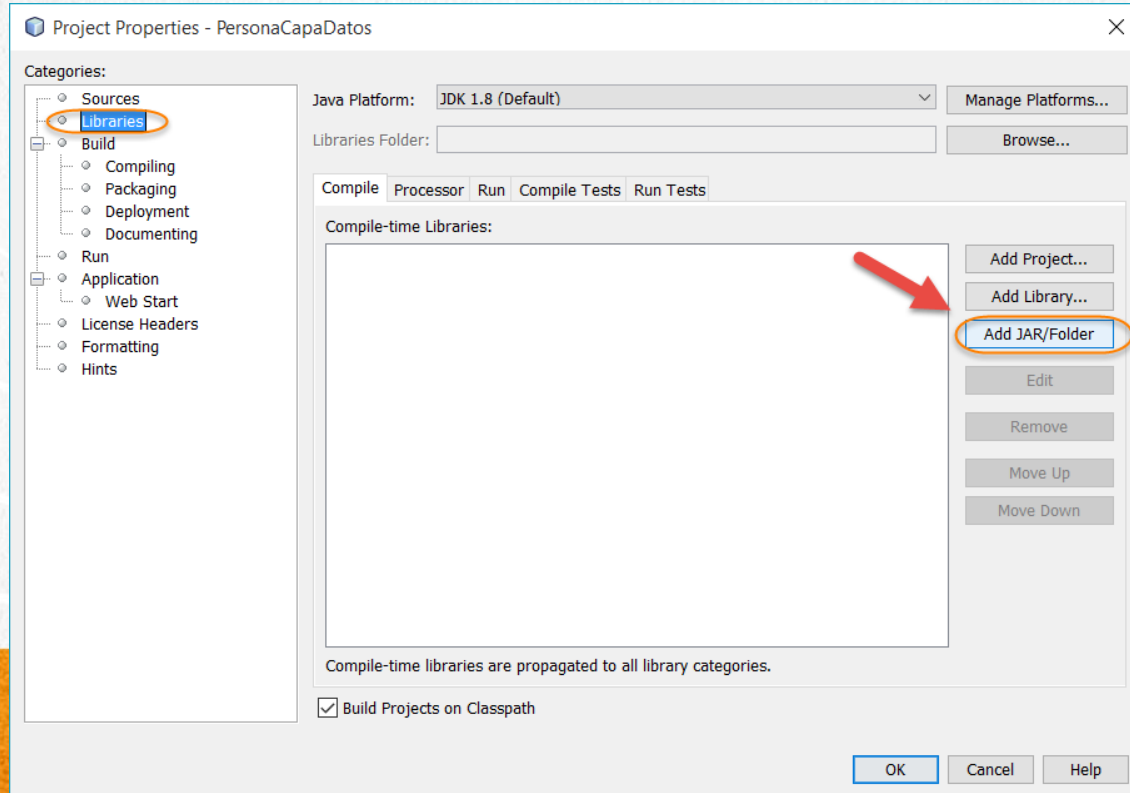
## PASO 2. ASIGNAR LIBRERIA

Vamos a asignar la librería de jdbc de mysql:



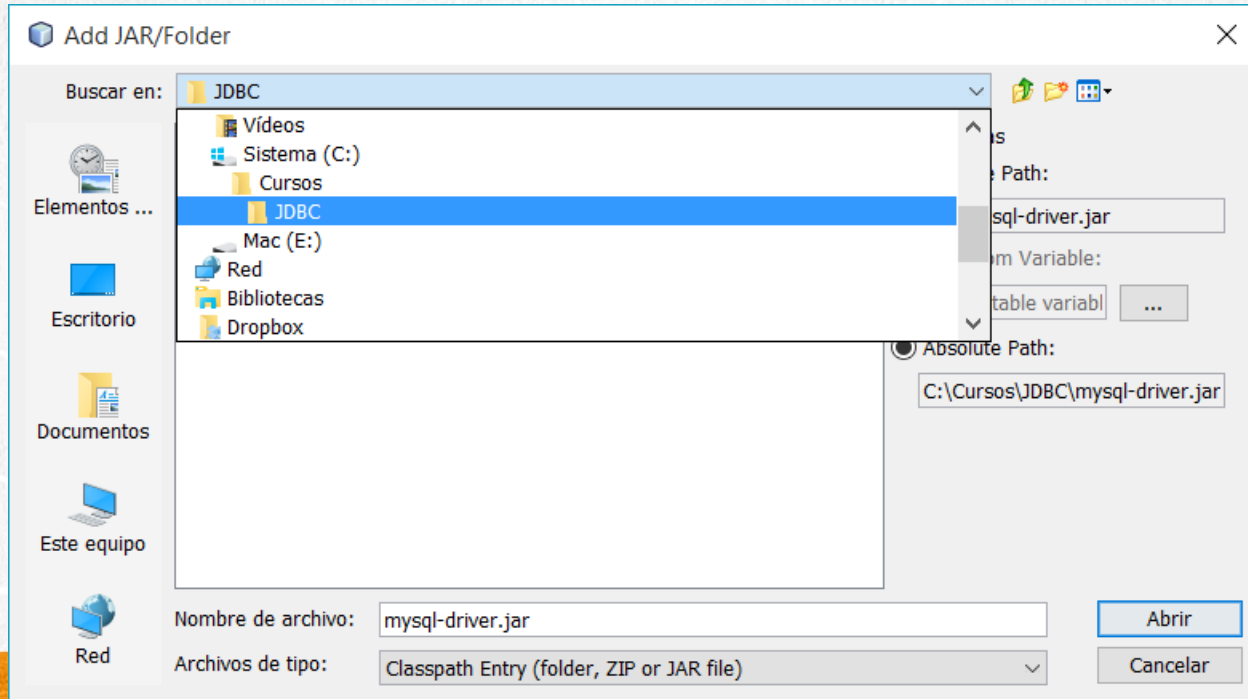
# PASO 2. ASIGNAR LIBRERIA (CONT)

Vamos a asignar la librería de jdbc de mysql:



## PASO 2. ASIGNAR LIBRERIA (CONT)

Vamos a asignar la librería de jdbc de mysql:

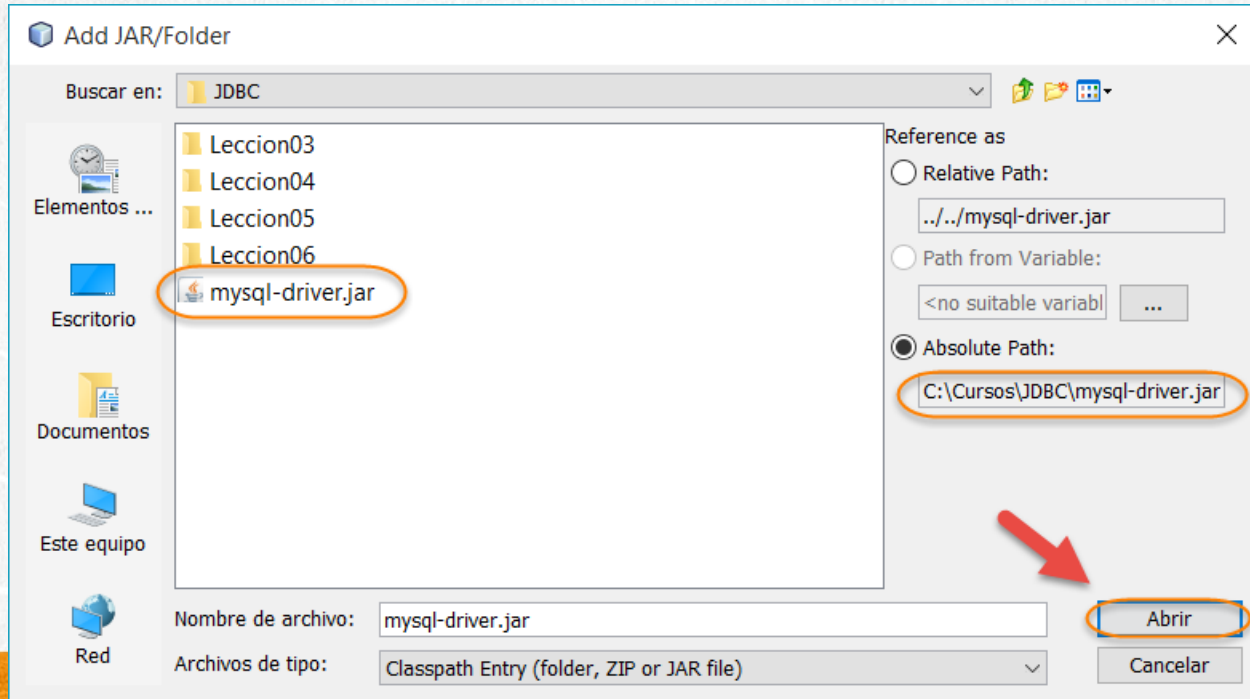


**CURSO DE JAVA CON JDBC**

[www.globalmentoring.com.mx](http://www.globalmentoring.com.mx)

## PASO 2. ASIGNAR LIBRERIA (CONT)

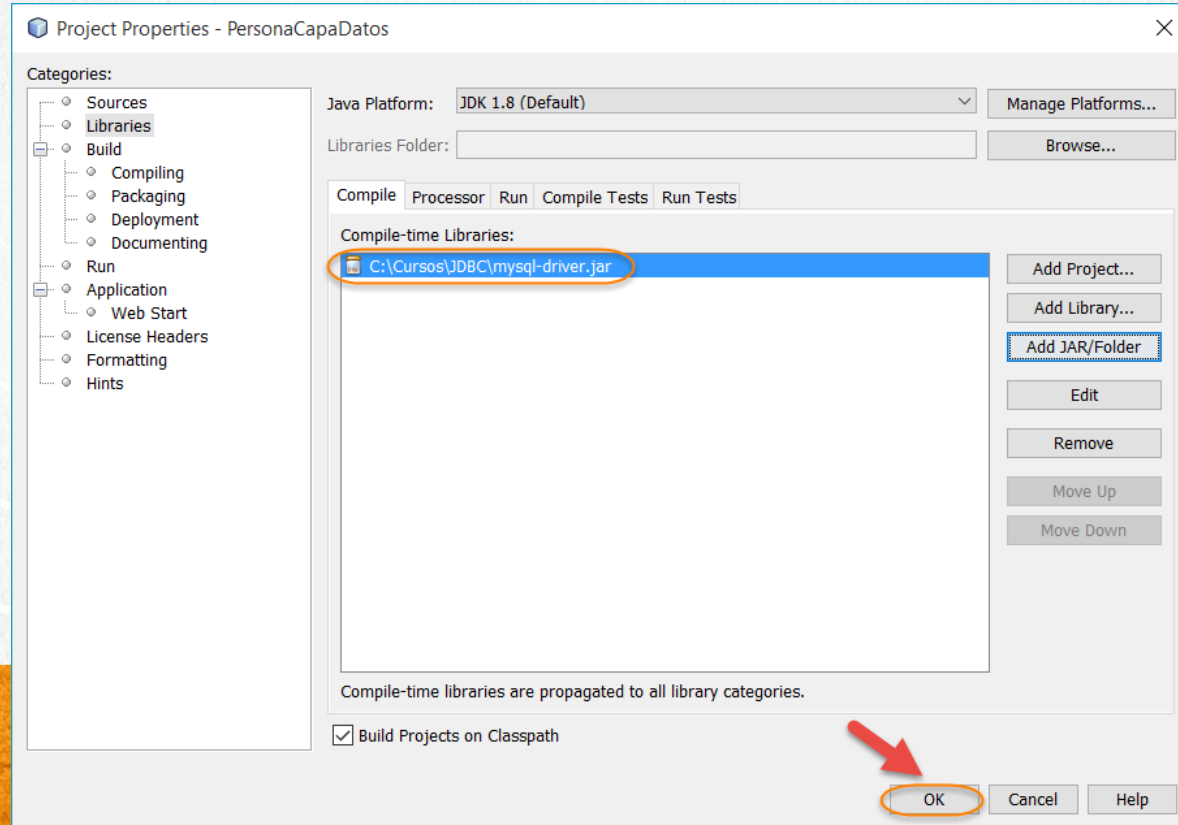
Vamos a asignar la librería de jdbc de mysql:





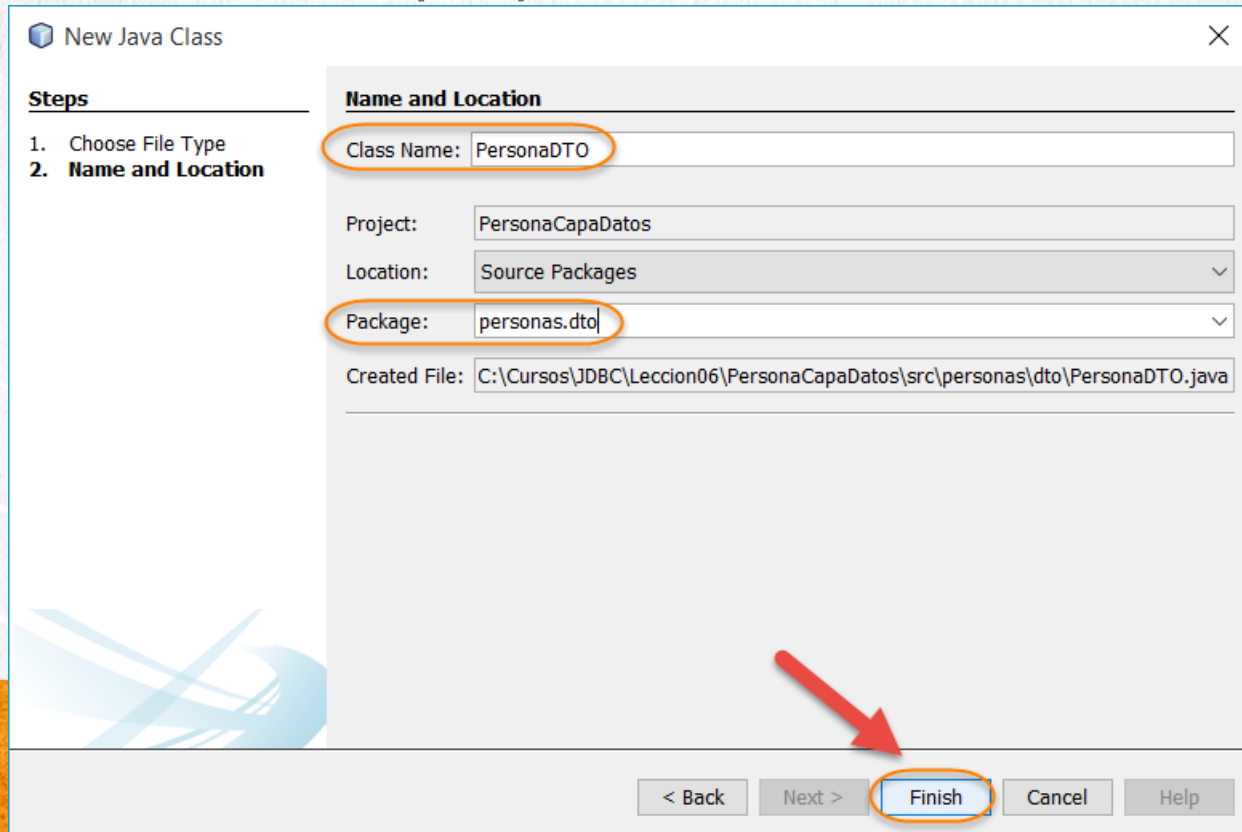
# PASO 2. ASIGNAR LIBRERIA (CONT)

Vamos a asignar la librería de jdbc de mysql:



# PASO 3. AGREGAMOS UNA CLASE

Agregamos una clase al proyecto:



**New Java Class**

**Steps**

1. Choose File Type
2. **Name and Location**

**Name and Location**

Class Name:

Project:

Location:

Package:

Created File:

# PASO 4. AGREGAMOS UNA CLASE

Agregamos una clase al proyecto:

New Java Class

**Steps**

1. Choose File Type
2. **Name and Location**

**Name and Location**

Class Name:

Project:

Location:

Package:

Created File:

< Back   Next >   **Finish**   Cancel   Help

# PASO 5. AGREGAMOS UNA CLASE

Agregamos una clase al proyecto:

**New Java Class**

**Steps**

1. Choose File Type
2. **Name and Location**

**Name and Location**

Class Name:

Project:

Location:

Package:

Created File:



# PASO 6. AGREGAMOS UNA CLASE

Agregamos una clase al proyecto:

New Java Class

**Steps**

1. Choose File Type
2. **Name and Location**

**Name and Location**

Class Name:

Project:

Location:

Package:

Created File:

< Back   Next >   **Finish**   Cancel   Help

# PASO 7. MODIFICAMOS EL CÓDIGO

## Archivo PersonaDTO.java:

```
package personas.dto;

public class PersonaDTO {

    public PersonaDTO() {
    }

    public PersonaDTO(int id_persona) {
        this.id_persona = id_persona;
    }

    private int id_persona;

    private String nombre;

    private String apellido;

    public int getId_persona() {
        return id_persona;
    }

    public void setId_persona(int idPersona) {
        id_persona = idPersona;
    }
```

```
    public String getNombre() {
        return nombre;
    }

    public void setNombre(String nombre) {
        this.nombre = nombre;
    }

    public String getApellido() {
        return apellido;
    }

    public void setApellido(String apellido) {
        this.apellido = apellido;
    }

    @Override
    public String toString() {
        return "Persona{" + "id_persona=" +
            id_persona + ", nombre=" + nombre + ", apellido=" +
            apellido + '}';
    }
}
```

# PASO 8. MODIFICAMOS EL CÓDIGO

## Archivo Conexion.java:

```
package personas.jdbc;
import java.sql.*;

public class Conexion {
    private static final String JDBC_DRIVER = "com.mysql.jdbc.Driver";
    private static final String JDBC_URL = "jdbc:mysql://localhost/sga?useSSL=false";
    private static final String JDBC_USER = "root";
    private static final String JDBC_PASS = "admin";
    private static Driver driver = null;

    public static synchronized Connection getConnection()
        throws SQLException {
        if (driver == null) {
            try {
                Class jdbcDriverClass = Class.forName(JDBC_DRIVER);
                driver = (Driver) jdbcDriverClass.newInstance();
                DriverManager.registerDriver(driver);
            } catch (Exception e) {
                System.out.println("Fallo en cargar el driver JDBC");
                e.printStackTrace();
            }
        }
        return DriverManager.getConnection(JDBC_URL, JDBC_USER, JDBC_PASS);
    }
}
```

# PASO 8. MODIFICAMOS EL CÓDIGO (CONT)

## Archivo Conexion.java:

```
public static void close(ResultSet rs) {  
    try {  
        if (rs != null) {  
            rs.close();  
        }  
    } catch (SQLException sqle) {  
        sqle.printStackTrace();  
    }  
}  
  
//Cierre del PreparedStatement  
public static void close(PreparedStatement stmt) {  
    try {  
        if (stmt != null) {  
            stmt.close();  
        }  
    } catch (SQLException sqle) {  
        sqle.printStackTrace();  
    }  
}
```

```
//Cierre de la conexion  
public static void close(Connection conn) {  
    try {  
        if (conn != null) {  
            conn.close();  
        }  
    } catch (SQLException sqle) {  
        sqle.printStackTrace();  
    }  
}
```



# PASO 9. MODIFICAMOS EL CÓDIGO

## Archivo PersonaDao.java:

```
package personas.jdbc;

import java.sql.SQLException;
import java.util.List;
import personas.dto.PersonaDTO;

/**
 * Esta interfaz contiene los métodos abstractos con las
 * operaciones básicas sobre la tabla de Persona
 * CRUD (Create, Read, Update y Delete)
 * Se debe crear una clase concreta para implementar el
 * código asociado a cada método
 * @author Ubaldo
 */
public interface PersonaDao {

    public int insert(PersonaDTO persona)
        throws SQLException;

    public int update(PersonaDTO persona)
        throws SQLException;

    public int delete(PersonaDTO persona)
        throws SQLException;

    public List<PersonaDTO> select() throws SQLException;
}
```

# PASO 10. MODIFICAMOS EL CÓDIGO

## Archivo PersonaDaoJDBC.java:

```
package personas.jdbc;

import java.sql.*;
import java.util.ArrayList;
import java.util.List;
import personas.dto.PersonaDTO;

/**
 * Esta clase implementa la clase PersonaDao es una implementacion con la
 * tecnologia JDBC podría haber otro tipo de implementaciones con tecnologias
 * como Hibernate, iBatis, SpringJDBC, etc.
 *
 * @author Ubaldo
 *
 */
public class PersonaDaoJDBC implements PersonaDao {

    private Connection userConn;

    private final String SQL_INSERT = "INSERT INTO persona(nombre, apellido) VALUES(?,?)";

    private final String SQL_UPDATE = "UPDATE persona SET nombre=?, apellido=? WHERE id_persona=?";

    private final String SQL_DELETE = "DELETE FROM persona WHERE id_persona = ?";

    private final String SQL_SELECT = "SELECT id_persona, nombre, apellido FROM persona";
```

# PASO 10. MODIFICAMOS EL CÓDIGO (CONT)

## Archivo PersonaDaoJDBC.java:

```
public PersonaDaoJDBC () {  
  
}  
  
public PersonaDaoJDBC(Connection conn) {  
    this.userConn = conn;  
}
```

```
/**  
 * El metodo insert recibe como argumento  
 * un objeto DTO el cual viene de  
 * otra capa, y se extraen sus valores para crear un nuevo registro  
 */  
@Override  
public int insert(PersonaDTO persona) throws SQLException {  
    Connection conn = null;  
    PreparedStatement stmt = null;  
    int rows = 0;  
    try {  
        conn = (this.userConn != null) ? this.userConn :  
Conexion.getConnection();  
        stmt = conn.prepareStatement(SQL_INSERT);  
        int index = 1;  
        stmt.setString(index++, persona.getNombre());  
        stmt.setString(index, persona.getApellido());  
        System.out.println("Ejecutando query:" + SQL_INSERT);  
        rows = stmt.executeUpdate();  
        System.out.println("Registros afectados:" + rows);  
    } finally {  
        Conexion.close(stmt);  
        if (this.userConn == null) {  
            Conexion.close(conn);  
        }  
    }  
  
    return rows;  
}
```

# PASO 10. MODIFICAMOS EL CÓDIGO (CONT)

## Archivo PersonaDaoJDBC.java:

```
@Override
public int update(PersonaDTO persona)
    throws SQLException {
    Connection conn = null;
    PreparedStatement stmt = null;
    int rows = 0;
    try {
        conn = (this.userConn != null) ? this.userConn :
Conexion.getConnection();
        System.out.println("Ejecutando query:" + SQL_UPDATE);
        stmt = conn.prepareStatement(SQL_UPDATE);
        int index = 1;
        stmt.setString(index++, persona.getNombre());
        stmt.setString(index++, persona.getApellido());
        stmt.setInt(index, persona.getId_persona());
        rows = stmt.executeUpdate();
        System.out.println("Registros actualizados:" + rows);
    } finally {
        Conexion.close(stmt);
        if (this.userConn == null) {
            Conexion.close(conn);
        }
    }
    return rows;
}
```

```
@Override
public int delete(PersonaDTO persona) throws SQLException {
    Connection conn = null;
    PreparedStatement stmt = null;
    int rows = 0;
    try {
        conn = (this.userConn != null) ? this.userConn :
Conexion.getConnection();
        System.out.println("Ejecutando query:" + SQL_DELETE);
        stmt = conn.prepareStatement(SQL_DELETE);
        stmt.setInt(1, persona.getId_persona());
        rows = stmt.executeUpdate();
        System.out.println("Registros eliminados:" + rows);
    } finally {
        Conexion.close(stmt);
        if (this.userConn == null) {
            Conexion.close(conn);
        }
    }
    return rows;
}
```



# PASO 10. MODIFICAMOS EL CÓDIGO (CONT)

## Archivo PersonaDaoJDBC.java:

```
@Override
public List<PersonaDTO> select() throws SQLException {
    Connection conn = null;
    PreparedStatement stmt = null;
    ResultSet rs = null;
    PersonaDTO personaDTO = null;
    List<PersonaDTO> personas = new ArrayList<PersonaDTO>();
    try {
        conn = (this.userConn != null) ? this.userConn : Conexion.getConnection();
        stmt = conn.prepareStatement(SQL_SELECT);
        rs = stmt.executeQuery();
        while (rs.next()) {
            int idPersonaTemp = rs.getInt(1);
            String nombreTemp = rs.getString(2);
            String apellidoTemp = rs.getString(3);
            personaDTO = new PersonaDTO();
            personaDTO.setId_persona(idPersonaTemp);
            personaDTO.setNombre(nombreTemp);
            personaDTO.setApellido(apellidoTemp);
            personas.add(personaDTO);
        }
    } finally {
        Conexion.close(rs);
        Conexion.close(stmt);
        if (this.userConn == null) {
            Conexion.close(conn);
        }
    }
    return personas;
}
```

# PASO 11. MODIFICAMOS EL CÓDIGO

## Archivo TestPersonas.java:

```
package personas.test;

import java.sql.SQLException;
import java.util.List;
import personas.dto.PersonaDTO;
import personas.jdbc.PersonaDao;
import personas.jdbc.PersonaDaoJDBC;

public class TestPersonas {

    public static void main(String[] args) {
        //Utilizamos el tipo interface como referencia
        //a una clase concreta
        PersonaDao personaDao = new PersonaDaoJDBC();

        //Creamos un nuevo registro
        //Hacemos uso de la clase persona DTO la cual se usa
        //para transferir la informacion entre las capas
        //no es necesario especificar la llave primaria
        //ya que en este caso es de tipo autonumerico y la BD se encarga
        //de asignarle un nuevo valor
        PersonaDTO persona = new PersonaDTO();
        persona.setNombre("mario");
        persona.setApellido("lopez01");
        //Utilizamos la capa DAO para persistir el objeto DTO
        try {
            //personaDao.insert(persona);

            //eliminamos un registro, el id 3
            //personaDao.delete( new PersonaDTO(3));
        }
    }
}
```

# PASO 11. MODIFICAMOS EL CÓDIGO (CONT)

## Archivo TestPersonas.java:

```
//actualizamos un registro
PersonaDTO personaTmp= new PersonaDTO();
//
//    personaTmp.setId_persona(2);//actualizamos el registro 2
//    personaTmp.setNombre("Mario");
//    personaTmp.setApellido("lopez02");
//    personaDao.update(personaTmp);

//Seleccionamos todos los registros
List<PersonaDTO> personas = personaDao.select();
for (PersonaDTO personaDTO : personas) {
    System.out.print( personaDTO );
    System.out.println();
}

} catch (SQLException e) {
    System.out.println("Excepcion en la capa de prueba");
    e.printStackTrace();
}

}
```

# PASO 12. EJECUTAMOS EL PROYECTO

PersonaCapaDatos - NetBeans IDE

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help

<default config>

Projects Files Services

- ManejoTransacciones
- PersonaCapaDatos
  - Source Packages
    - personas.dto
      - PersonaDTO.java
    - personas.jdbc
      - Conexion.java
      - PersonaDao.java
      - PersonaDaoJDBC.java
    - personas.test
      - TestPersonas.java
  - Libraries
    - mysql-driver.jar
    - JDK 1.8 (Default)

TestPersonas.java

```
36     personaTmp.setApellido("lopez02");
37     personaDao.update(personaTmp);
38     */
39
40     //Seleccionamos todos los registros
41     List<PersonaDTO> personas = personaDao.select();
42     for (PersonaDTO u : personas) {
43         System.out.print(" id_persona:" + u.getId_persona());
44         System.out.print(" persona:" + u.getNombre());
45         System.out.print(" password:" + u.getApellido());
46         System.out.println();
47     }
48
49     } catch (SQLException e) {
50         System.out.println("Excepcion en la capa de prueba");
51         e.printStackTrace();
52     }
```

Find What: getPassword Previous Next

Replace With: getApellido Replace Replace All Replace Backwards Preserve Case

personas.test.TestPersonas main try

Output - PersonaCapaDatos (run)

```
run:
id_persona:2 persona:Regreso2 password:Regreso
id_persona:3 persona:Alberto password:Juarez
id_persona:4 persona:Miguel2 password:Ayala2
id_persona:5 persona:mario password:lopez01
BUILD SUCCESSFUL (total time: 1 second)
```



# CONCLUSIÓN DEL EJERCICIO

Como este ejercicio hemos visto como crear una capa de datos utilizando JDBC.

Además aplicamos algunos patrones de diseño como son: DAO y DTO, los cuales estaremos utilizando cuando creemos nuestras capas de datos.



Experiencia y Conocimiento para tu vida

**CURSO DE JAVA CON JDBC**

[www.globalmentoring.com.mx](http://www.globalmentoring.com.mx)

**CURSO ONLINE**

# **JAVA CON JDBC**

---

Por: Ing. Ubaldo Acosta



Experiencia y Conocimiento para tu vida

**CURSO DE JAVA CON JDBC**

[www.globalmentoring.com.mx](http://www.globalmentoring.com.mx)